Project 1
Training Neural Networks

Name	ID	DEPARTMENT
Ahmed Mostafa	20221372883	AI
Mazen Gaber	20221372110	AI

Introduction

The goal of this project is to develop a handwritten digit recognition system using neural networks.
Dataset used: The MNIST which contains 60,000 training examples and labels. Each example consists of 784 pixel values representing a 28x28 image of a handwritten digit (0-9) downloaded from kaggel

Step 1: Data Preprocessing and Splitting

In this project, several preprocessing steps were performed on the training data before splitting it into a training set and a validation set. The purpose of these preprocessing steps is to prepare the data for training a neural network.

1. Splitting into Features and Labels:

The original dataset contains 785 values per example, where the first value represents the label (the digit from 0 to 9) and the remaining 784 values represent the pixel

values. To feed the data into the neural network, we will separate the features (pixel values) and labels.

2. Normalization:

The pixel values in the MNIST dataset range from 0 to 255, representing the intensity of each pixel. To make the features have a similar scale we will normalize the data. Normalization means scaling pixel values between 0 and 1 or -1 and 1

3. Random Shuffling:

Shuffling the data randomly is important so that the order of the examples does not bias in the training. Shuffling will randomize the data and remove any biases that exists in the original data. This makes the model generalize better.

4. Splitting into Training and Validation Sets:

After preprocessing, now we will split the data into 80% training set and 20% validation set.The training set is used to train the neural network, while the validation set is used to evaluate the model's performance during training.

Step 2: Building the Neural Network Architecture:

A neural network architecture is constructed using the PyTorch library. The architecture consists of at least two hidden layers with ReLU activation functions. The number of hidden layers and the number of neurons in each layer

can be adjusted based on our preferences. The model is trained using the Adam optimizer, a learning rate of 0.01, and the cross-entropy loss function.

1. Designing the Neural Network Architecture:

In this step, we designed the architecture of the neural network. The architecture includes the number of hidden layers, the number of perceptrons in each layer, and the activation function used between the layers.

2. Changing the Number of Hidden Layers:

Adding more hidden layers will make the model learn more complex patterns in the data and achieve better performance.

Adding too many hidden layers can cause overfitting.

3. Changing the Number of Perceptrons in Each Layer:

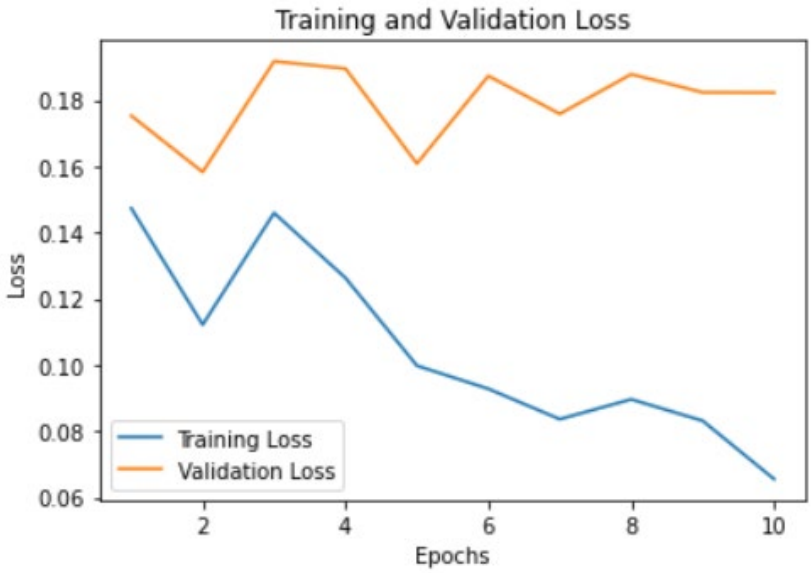
Adding more perceptrons increases the number parameters that can be learned.

Adding too many perceptrons increases the computational complexity and can cause overfitting.

The training and validation loss, as well as the training and validation accuracy, are plotted to monitor the model's performance.

Training / Validation Loss :

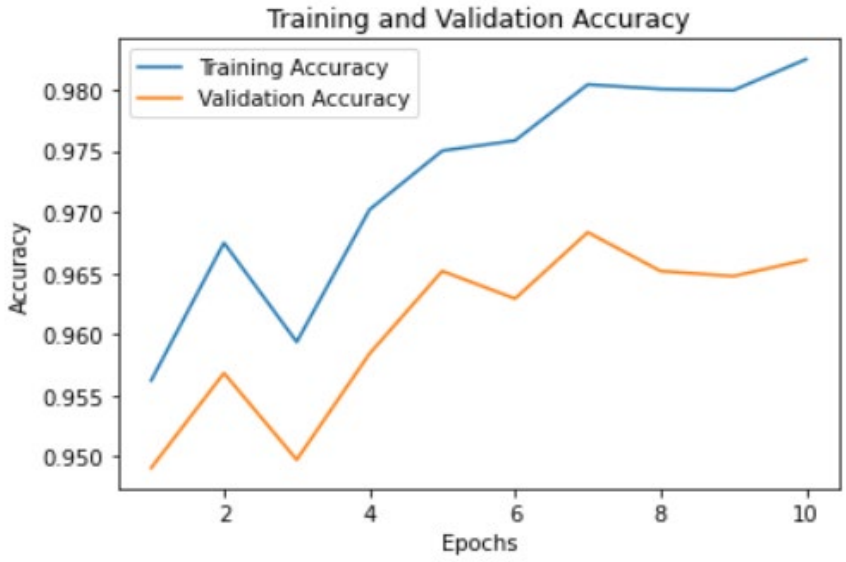
This plot shows the change in loss over epochs during the training process. The loss represents the error between the predicted and actual values.



- As the epochs progress, both the training and validation loss decrease.
- The training loss begins by decreasing rapidly then the decrease rate starts to get reduced.
- The validation loss also decreases but at a slower rate compared to the training loss.

Training / Validation Accuracy :

This plot shows the accuracy of the model on the training and validation sets over epochs.



- The training accuracy gradually increases as the model learns from the training data.
- The validation accuracy also increases, but at a slower rate compared to the training accuracy.
- After a certain number of epochs the accuracy of the training and validation stops improving.

From these observations we can say that:

- The decreasing loss and increasing accuracy means that the model is learning and improving its performance over epochs.

- The smaller the gap between the training and validation loss/accuracy, the better the model's generalization capability.
- The slow increase in validation accuracy means that the model is reaching its optimal performance on the dataset.

Step 3: Adding Dropout and Layer Normalization:

Dropout and Layer Normalization layers are added to the neural network architecture.

Dropout randomly sets a number of input units to zero during training, to prevent overfitting.
Layer Normalization normalizes the outputs of each layer, to improve generalization.

1. Dropout:

When we added dropout with probability 0.2 or 0.4, this resulted in small reduction in training accuracy but improved validation accuracy which means less overfitting.

But when we changed the probability of dropout to 0.6 this resulted in underfitting, this means that we applied too much dropout.

2. Layer Normalization:

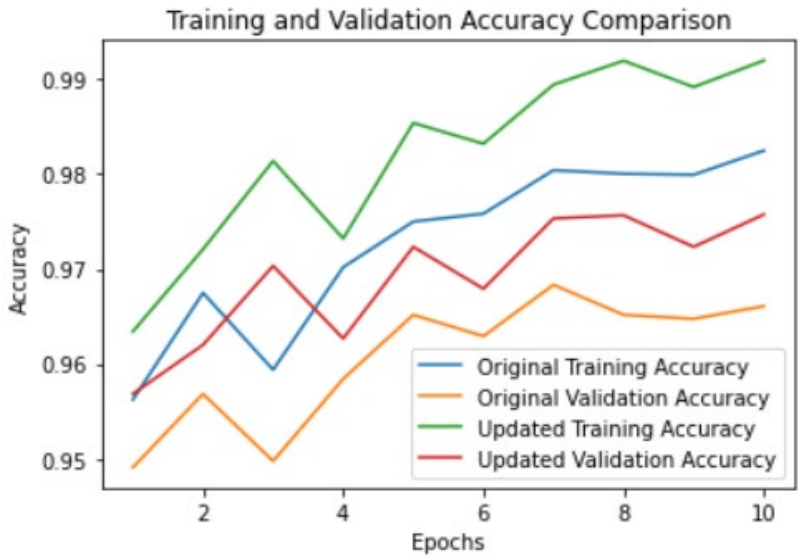
Layer normalization didn't have a big change on the performance. There was an improvement in validation accuracy but the change was not as big as with dropout.

The training and validation loss/accuracy plots are compared with the original model to see what happened to the model performance

Train / Validation loss:

- The training loss and the validation loss starts relatively high and gradually decreases over epochs.
- The loss values for both training and validation starts to be constant after a certain number of epochs.

Train / Validation Accuracy



- The training accuracy starts low and increases as the model learns from the training data.
- The validation accuracy was the same, it started low then started to increase.
- The accuracy values for both training and validation begins to be constant after a certain number of epochs.

From these observations we could indicate the following:

- The decreasing loss and increasing accuracy means that the model is learning and improving its performance over epochs.
- The convergence of training and validation loss and accuracy means that the model is not overfitting or underfitting.

- The not changing loss and accuracy values after a certain number of epochs means that the model has reached a constant state and training it more will not have any improvements.

Step 4: Training Multiple Models with Varying Hyperparameters:

We will train 3 models with different values of Learning rates and different valus of Dropout

a. Learning Rate: Three learning rates (0.001, 0.0005, 0.0001) are used.

b. Probability of Dropout: Three dropout probabilities (0.2, 0.4, 0.6) are used.

1. Dropout Comparison:

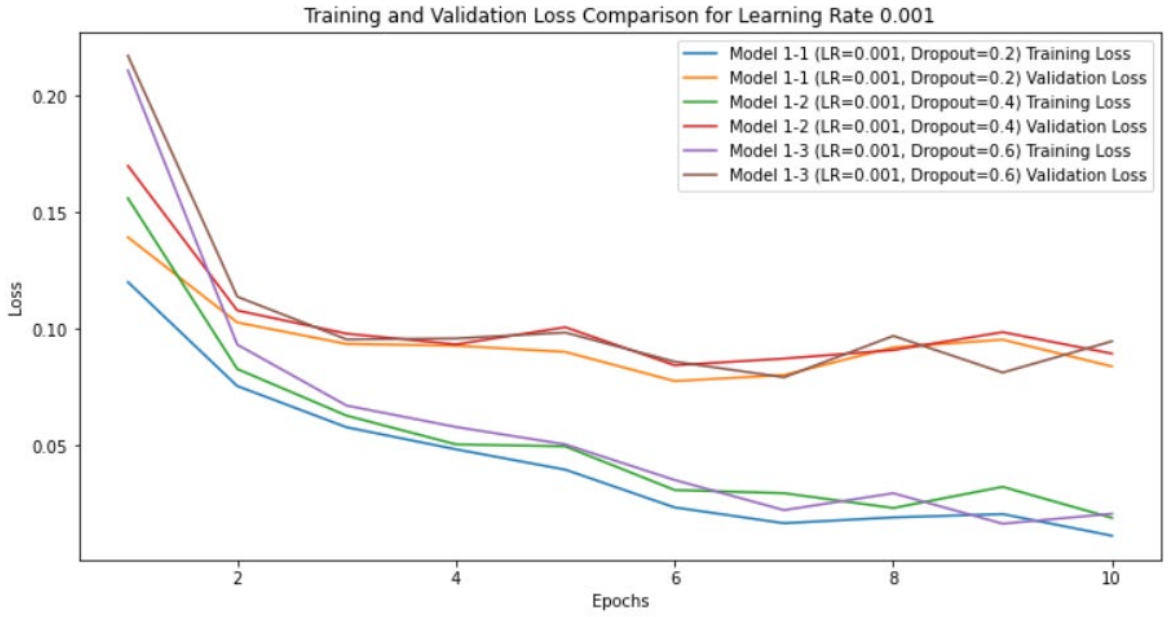
- Values like 0.4 and 0.6 are high dropout values, this means a large number of input units will be randomly dropped during training
- 0.2 is a low value, which means fewer number of input units will be dropped.
- We compared the training and validation loss as well as the training and validation accuracy across these different dropout probabilities.

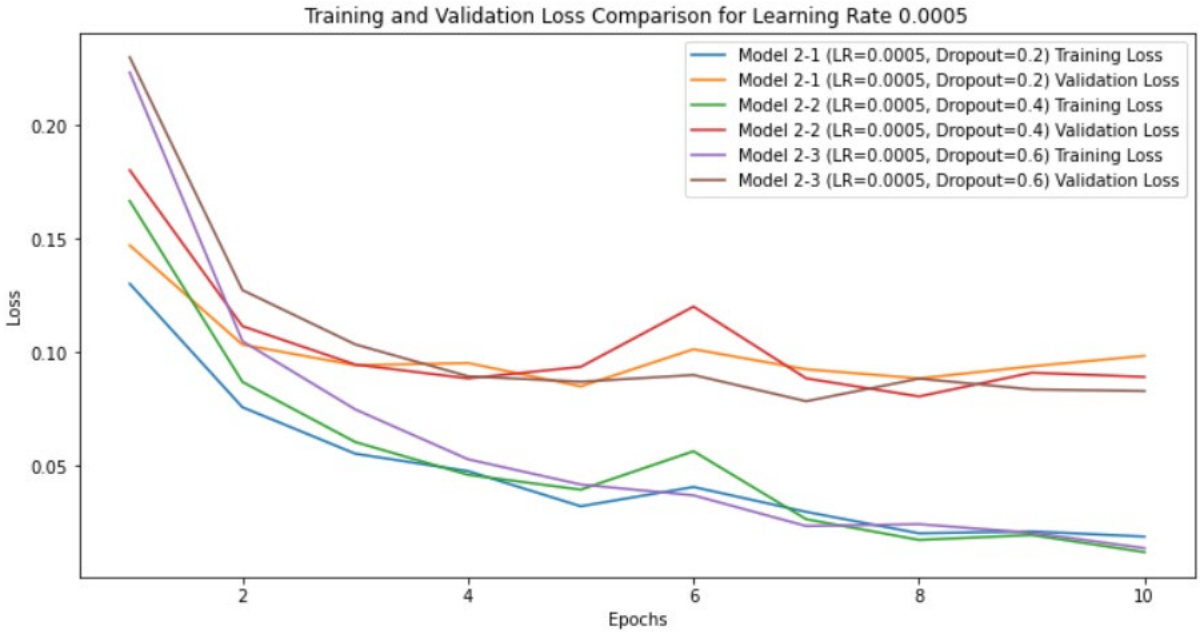
2. Layer Normalization Comparison:

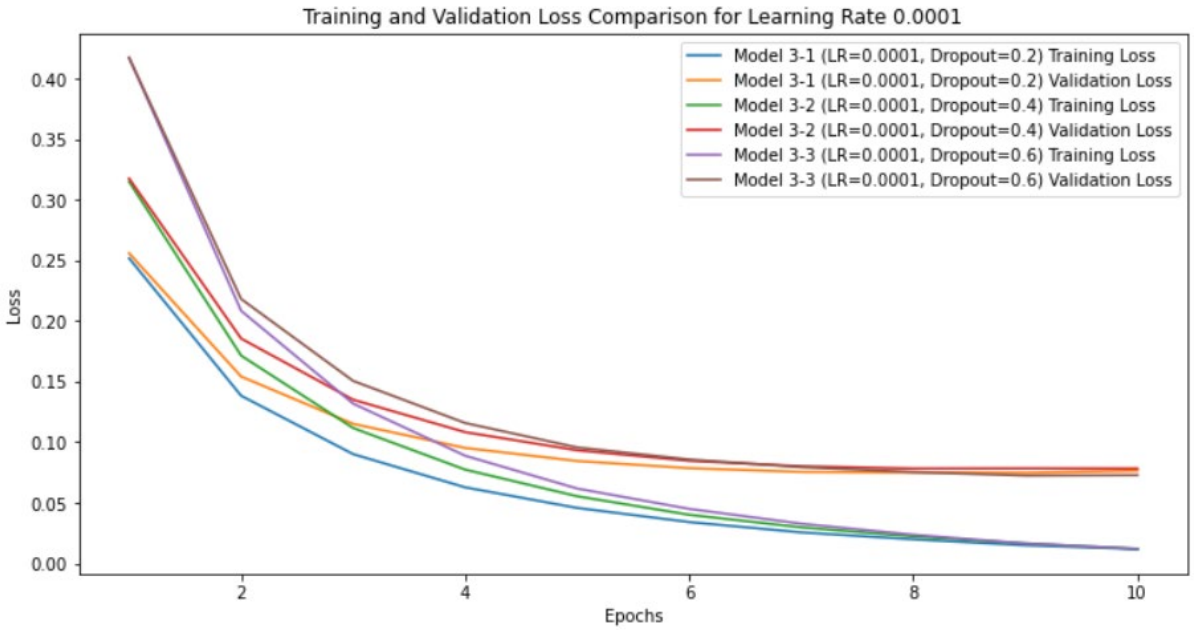
All the models in step 4 included Layer Normalization.

The training and validation loss/accuracy plots are generated for each model, allowing a comparison of their performance.

Train / Validation loss:

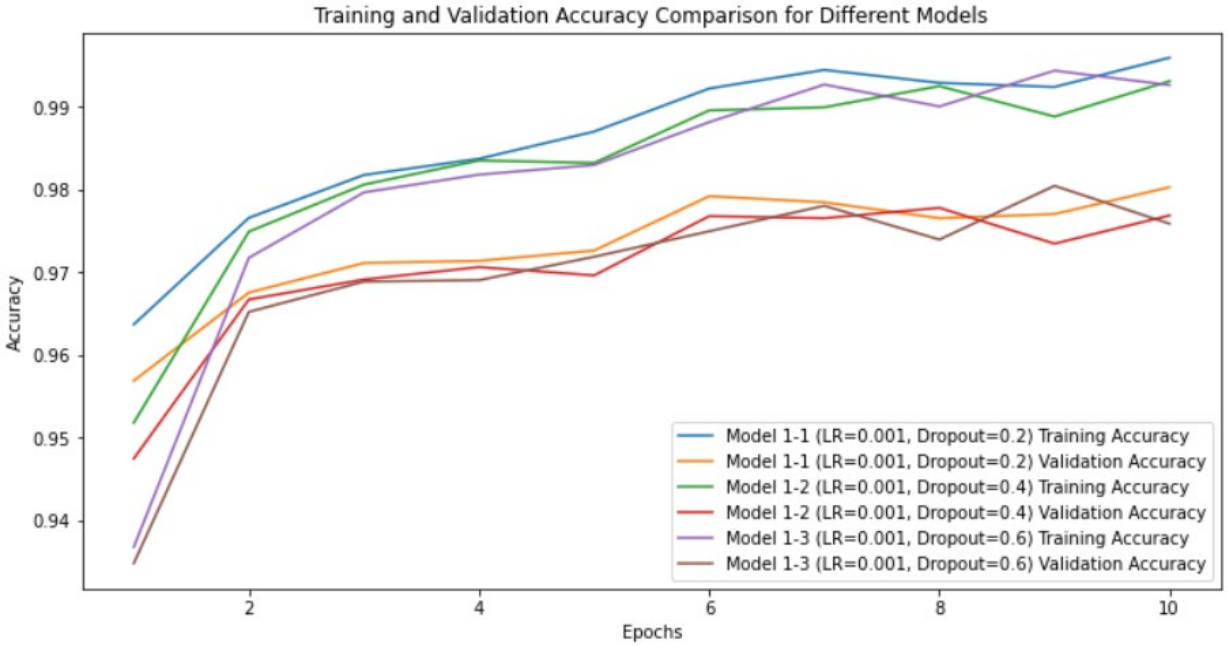


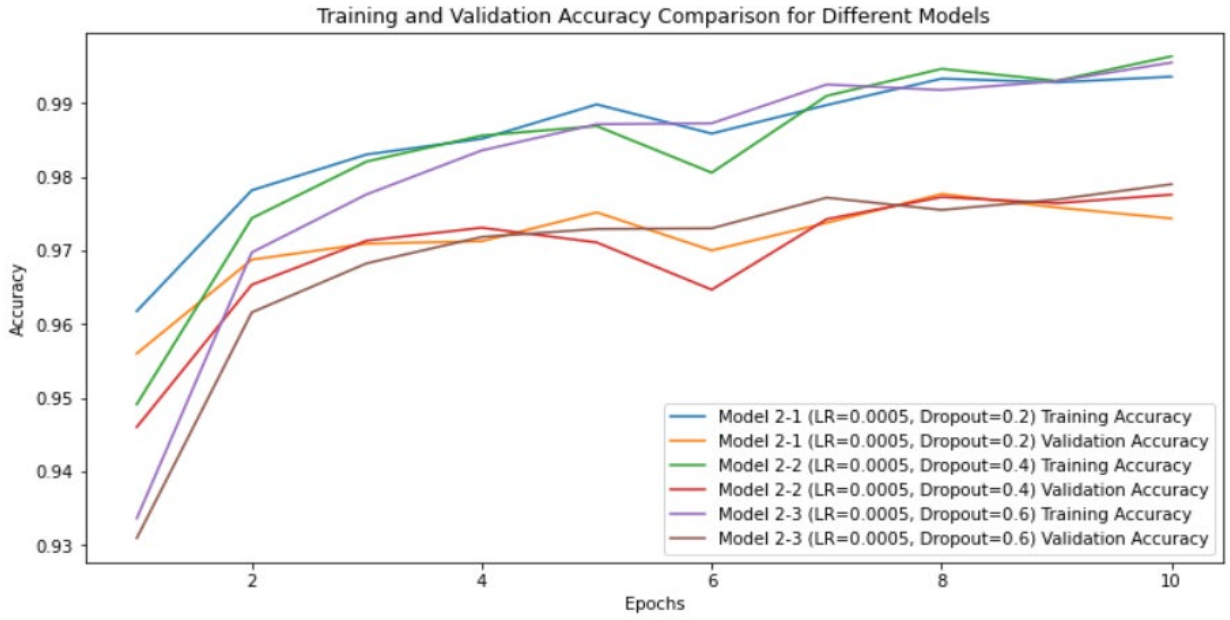


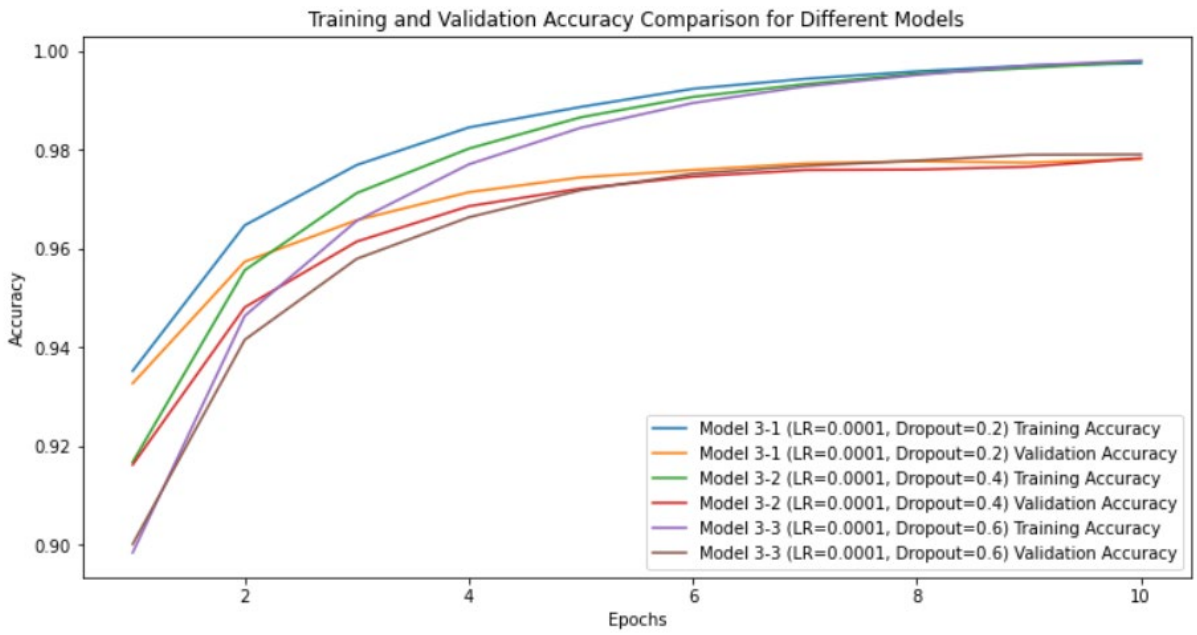


- Models with learning rate of 0.001 have lower training and validation loss compared to models with learning rate 0.0005 and 0.0001
- Models with dropout probability 0.4 have the lowest training and validation loss, then the models with probability 0.6 and 0.2
- Overall, the models with a learning rate of 0.001 and dropout probability of 0.4 perform the best in terms of minimizing the loss.

Train / Validation Accuracy







- Models with learning rate 0.001 have higher training and validation accuracy compared to the models with learning rates 0.0005 and 0.0001.
- Models with dropout probability 0.4 have the highest training and validation accuracy, then the models with probability 0.6 and 0.2
- Similar to the loss comparison, the models with a learning rate of 0.001 and dropout probability of 0.4 has best accuracy.

So, overall based on the results, it appears that using a learning rate of 0.001 and a dropout probability of 0.4 has the best performance in both minimizing the loss and maximizing the accuracy.